Robust Hypothesis Testing of Location Parameters using Lq-Likelihood-Ratio-Type Test in Python

by

Anton Alyakin

A thesis submitted to The Johns Hopkins University
in conformity with the requirements for the degree of
Master of Science in Engineering

Baltimore, Maryland
November 2019

Abstract

A t-test is considered a standard procedure for inference on population means and is widely used in scientific discovery. However, as a special case of a likelihood-ratio test, t-test often shows drastic performance degradation due to the deviations from its hard-to-verify distributional assumptions. Alternatively, we propose a new two-sample Lq-likelihood-ratio-type test (LqRT) along with an easy-to-use Python package implementing it. LqRT preserves high power when the distributional assumption is violated, and maintains the satisfactory performance when the assumption is valid. As numerical studies suggest, LqRT dominates other robust tests, such as Wilcoxon test and sign test, in power, while maintaining a valid size. To the extent that the robustness of the Wilcoxon test (minimum asymptotic relative efficiency (ARE) of the Wilcoxon test vs the t-test is 0.864) suggests that the Wilcoxon test should be the default test of choice (rather than "use Wilcoxon if there is evidence of non-normality," the default position should be "use Wilcoxon unless there is good reason to believe the normality assumption"), the results in this thesis suggest that the LqRT is potentially the new default go-to test for practitioners.

Thesis Advisor: Carey E. Priebe

Acknowledgments

I would like to thank my advisor Carey Priebe for supporting the work presented here. I am indebted to him for his contribution to my formation both as a researcher and a human being. His enthusiasm and wiseness are unparalleled and are a continious source of my motivation. Working with him made me finally feel that my voice is being heard.

I would like to thank the entire JHU Applied Mathematics and Statistics Department for my challenging, yet rewarding undergraduate and graduate experiences. A special meniton goes to Avanti Athreya, who has been my advisor and mentor throughout this time. Over these years, I have learned a lot from her, ranging on the topics from statistical theory to pedagogical practice.

I would like to thank Yichen Qin, the author of the original LqRT paper, for his contributions to this work which included guidance on understanding LqRT and extensive editorial help with the final manuscript. I was standing on the shoulders of giants while working on this project.

Finally, I am very grateful to my friends and family for their help and guidance, and I am forever indebted to my Mom, my Dad and my sister Anna, for their unconditional support and encouragement.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Classical testing procedures, such as t-test, claim its optimal performance and its highest power while relying heavily on the assumptions that are neither verifiable nor actually hold in many real data settings. For example, datasets often include outliers, observations that deviate significantly from other elements of the sample, and are unlikely under the assumed underlying distribution (Grubbs, 1969). Outliers' presence may be due to human error, instrument malfunction or the complexity of the true data-generating process compared to the model (Qin and Priebe, 2017). In the presense of outliers, the validity of the classical test is undermined and its power is severely degraded. Therefore, it is important to develop statistical hypothesis testing procedures that are not significantly affected by their presence.

An ideal robust method should show nearly optimal performance when outliers are present, or more generally, under the deviations from distributional assumptions (Hampel et al., 2011), and maintains satisfactory performance when assumptions are valid. For estimation, such examples include median or trimmed mean as measures of central tendency. For hypothesis testing, there has been relatively less investigation.

Historically, robust testing procedures are predominantly non-parametric. The first significance test used was a robust test, when in 1710 John Arbuthnot used a sign test to conclude that the probability of male and female births are not exactly equal (Conover, 1999, pp. 157-176) Other robust non-parametric testing procedures include Wilcoxon signed-rank test (Wilcoxon, 1945) and Wilcoxon rank-sum test, also known as Mann-Whitney U test (Mann and Whitney, 1947). Parametric robust tests are less common because they naturally depend on the parametric distribution assumptions. The most notable example of such is the Huber Ratio Test (Huber, 1965).

Recently, Qin and Priebe, 2017 have proposed a robustified likelihood ratio test by leveraging the idea of Lq-likelihood (Ferrari and Yang, 2010), which they call Lq-likelihood-ratio-type test (LqRT). By assigning observations different weights according to their likelihoods under the assumed model, LqRT is able to attain powers similar to parametric tests when model assumptions are valid, and maintain satisfactory powers similar to or higher than nonparametric tests when assumptions are violated, leading to a uniform domination over Wilcoxon test and sign test.

In the literature, it has been shown that the minimum asymptotic relative efficiency (ARE) of the Wilcoxon test vs the t-test is 0.864 (Hodges and Lehmann, 1956). which advocates that the Wilcoxon test should be the default test of choice as opposed to t-test, and that the practitioners should "use Wilcoxon unless there is good reason to believe the normality assumption," as opposed to "use Wilcoxon if there is evidence of non-normality." The desirable results of LqRT suggest that LqRT is potentially the new default go-to test for practitioners.

However, the original LqRT paper focuses mostly on the theoretical properties and is lack of easy implementation of the test for practitioners. In addition, it also misses many important cases such as two-sample paired/unpaired tests for the equivalence of locations, which are frequently encountered in scientific discovery. To unleash the power of LqRT, in this work, we present the extension of LqRT to the two-sample case, propose a new approach to select the tuning parameter q, more importantly, provide **LqRT**, a Python package which implements the one-sample, the two-sample paired and the two-sample unpaired LqRT, so that practitioners can easily adopt the proposed test in their analysis.

Python (Python Software Foundation, 2001–) is an interpreted, general-purpose programming language. There are a variety of statistical significance tests implemented in Python. The scientific computing package **SciPy** (Jones, Oliphant, Peterson, et al., 2001–) implements one-sample t-test as `scipy.stas.ttest_1samp()`, two-sample paired as `scipy.stas.ttest_rel()`, and two-sample unpaired as `scipy.stas.ttest_ind()`. It also contains non-parametric Wilcoxon Rank-sum and Wilcoxon Signed-Rank tests, implemented as `scipy.stats.ranksums()` and `scipy.stats.wilcoxon()`, respectively. **LqRT** uses a syntax that is similar to that of **SciPy**. There is no sign test in **SciPy**, but it is implemented as `statsmodels.stats.descriptivestats.sign_test()` in **statsmodels** (Seabold and Perktold, 2010). The proposed **LqRT** is a natural complement to the existing toolbox. As shown in our simulation studies,

LqRT offers much improved numerical performance while requires little modification of analysts' existing code.

The rest of the works is organized as follows. In Chapter 2 we present the LqRT procedure. Specifically, we first review the foundations of the Lq-likelihood, and then present the general form of LqRT. We further extend it to one-sample, paired two-sample and unpaired two-sample tests. Finally, we discuss the method to estimate the p-value and the tuning parameter q. We present the Python package, **LqRT**, and its core functions and their instructions in Chapter 3. In Chapter 4 we demonstrate the performance of **LqRT** compared to other tests implemented in Python. We conclude and discuss our results and possible future work in Chapter 5. Additional information such as the pseudo-codes are included in the Appendix.

Chapter 2

Lq-likelihood-ratio-type test

2.1 Preliminaries

Let $\boldsymbol{x} = [x_1, \ldots, x_n]^T$ denote a sample of independent observations from a hypothesized distribution $f(\cdot|\boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$. The log-likelihood of this sample is $\sum_{i=1}^{n} \log(f(x_i|\boldsymbol{\theta}))$, while the L$q$-likelihood (Ferrari and Yang, 2010) of the sample is defined as

$$\sum_{i=1}^{n} L_q\left(f\left(x_i|\boldsymbol{\theta}\right)\right),$$

where $L_q(\cdot)$ is a q-deformed logarithm first introduced in Tsallis, 1994. Specifically, for any positive u,

$$L_q(u) = \begin{cases} \log(u) & \text{if} \quad q = 1 \\ \frac{u^{1-q}-1}{1-q} & \text{otherwise} \end{cases}.$$

This function is equivalent to the Box-Cox transformation under a $\lambda = 1 - q$ reparameterization (Box and Cox, 1964). Note that $L_q(u) \to \log(u)$ as $q \to 1$. Therefore Lq-likelihood includes log-likelihood as speical case $q = 1$. The maximum Lq-likelihood estimation (MLqE) of $\boldsymbol{\theta}$, $\widehat{\boldsymbol{\theta}}$, can be naturally defined as the maximizer of Lq-likelihood (Ferrari and Yang, 2010),

$$\widehat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} L_q \left(f \left(x_i | \boldsymbol{\theta} \right) \right)$$

By solving for the root of the gradient of the Lq-likelihood, MLqE satifies

$$\begin{aligned} \mathbf{0} &= \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{n} L_q \left(x_i | \boldsymbol{\theta} \right) \Bigg|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \\ &= \sum_{i=1}^{n} \frac{\nabla_{\boldsymbol{\theta}} f \left(x_i | \boldsymbol{\theta} \right)}{f \left(x_i | \boldsymbol{\theta} \right)} f \left(x_i | \boldsymbol{\theta} \right)^{1-q} \Bigg|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \\ &= \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log f \left(x_i | \boldsymbol{\theta} \right) w_i \Bigg|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \end{aligned}$$

where $w_i := f \left(x_i | \boldsymbol{\theta} \right)^{1-q}$. Thus, it is a weighted version of the gradient of the log-likelihood where the weights are likelihoods taken to the power of $1 - q$. Clearly, for $q = 1$, the weights are all 1, so the MLqE coincides with the MLE. For $q < 1$, this reweighting allows to reduce the effect of potential outliers whose likelihood tend to be small. From hereafter, we formalize the notion of an outlier to such points with small likelihood. With the preliminaries introduced above, we are ready to introduce our proposed hypothesis testing procedure.

2.2 Lq-likelihood-ratio-type test

Suppose we have a sample $\boldsymbol{x}$ with a hypothesized density $f(x_i|\theta)$ and we are interested in testing the parameter θ of this density for $H_0 : \theta \in \Theta_0$ against $H_1 : \theta \in \Theta_1$. Assume that this parameter θ is a location parameter of a symmetric density f. Then we define the Lq-likelihood-ratio-type (LqRT) as follows: The test statistic is

$$D_q(\boldsymbol{x}) = 2 \sup_{\theta \in \Theta_0 \cup \Theta_1} \left\{ \sum_{i=1}^{n} L_q\left(f(x_i|\theta)\right) \right\} - 2 \sup_{\theta \in \Theta_0} \left\{ \sum_{i=1}^{n} L_q\left(f(x_i|\theta)\right) \right\},$$

and we reject the null hypothesis for the large values of D_q. This test falls in the category of the likelihood-ratio-type test, defined in Heritier and Ronchetti, 1994. Under some regularity conditions, the asymptotic distribution of the test statistic is a χ^2 for $q \in [0, 1]$. For $q = 1$, LqRT coincides with likelihood ratio test (LRT). For $q < 1$, LqRT has been shown to be more robust to the contamination than LRT, both theoretically and experimentally.

The robustness of LqRT comes from the fact the Lq-likelihood function is less sensitive to outliers than the log-likelihood function, but remains approximately as sensitive to the rest of the data points as the log-likelihood function. When the model assumptions is valid and data is clean, the LqRT behaves similarly to LRT. The test statistic follows a χ^2 distribution under the null and a non-central χ^2 distribution under the alternative. When the data is contaminated with the gross error model, the test statistic of LRT becomes an inflated χ^2 distribution (or inflated non-central χ^2) where the inflation magnitude depends the level of contamination, leading to a much large overlap between the null and alternative distributions and power degradation. On the other hand, the test statistic D_q of LqRT is much less affected by the contamination with very little inflation, maintaining a small overlap between the null and alternative distributions and a relatively high power (see Figure 2 in Qin and Priebe (2017)).

The inflation of LRT in this case is mainly due to the extreme values of the log-likelihood of the outliers. For example, an outlier $x_{outlier}$ under the assumed model $f(\cdot|\theta)$ usually takes a small likelihood, i.e., $f(x_{outlier}|\theta) \approx 0$. Taking the logarithm of a small likelihood, $\log f(x_{outlier}|\theta)$, leads to a large negative value, which inflates the test statistic. Alternatively, for LqRT, Lq function is bounded from below (i.e., $L_q(u) > -1/(1-q)$) which offers the robustness and maintains a similar shape to log which offers the high power. Because of this property, as shown later in the simulation, the LqRT is able to main high power similar to the parametric tests such as t test when the model assumption is valid and maintain high power similar to the nonparametric tests such as Wilcoxon test and sign test. Please refer to Qin and Priebe, 2017 for a more detailed discussion of asymptotic properties of LqRT.

Note that the test statistic D_q is the difference of the maximums of the Lq-likelihoods under the null hypothesis and under the union of null and alternative hypothesis. Therefore, to compute D_q, it is equivalently to obtain the MLqE under the null and alternative hypothesis However, there is no closed form solution to the maximization of for the Lq-likelihood function. Ferrari, 2008 suggests using an iterative mixture model EM-like re-weighting algorithm. Similarly to a regular EM, this algorithm arises naturally from the fact that the estimator would be easy to compute if the weights were known, whilst the weights themselves are easy to compute as a function of the data and parameters (Murphy, 2013). The general iterative reweighting algorithm for obtaining MLqE and the maximum of Lq-likelihood is summarized in Algorithm 1.

Algorithm 1 Iterative re-weighting algorithm for compute maximum of Lq-likelihood

function MLQE($\boldsymbol{x}$, q)
  $n \leftarrow$ LENGTH($\boldsymbol{x}$)

  # Initialize Parameters with MLE
  $\hat{\theta}^{(0)} \leftarrow MLE(\boldsymbol{x})$

  # Iterative Re-weighting
  for $s = 1, 2, \ldots$ *until convergence* **do**

    # Update Weights
    for $i \leftarrow 1, 2, \ldots, n$ **do**
      $w_i^{(s)} \leftarrow f(x_i|\hat{\theta}^{(s-1)})^{1-q}$
    end for

    # Estimate the parameters using updated weights
    $\hat{\theta}^{(s)} \leftarrow$ the root of $0 = \sum_{i=1}^{n} w_i^{(s)} \frac{\partial}{\partial\theta} \log f(x_i|\theta)$

  end for
  return $\hat{\theta}^{(s)}$ as MLqE and $\sum_{i=1}^{n} L_q(f(x_i|\hat{\theta}^{(s)}))$ as the maximum of Lq-likelihood.
end function

Note that MLqE is generally neither consistent nor asymptotically unbiased. There have been various methods proposed to modify the procedure in the way to make it consistent for all problems. This includes using sequences q_n with a property $q_n \to 1$ (Ferrari and Yang, 2010) and correcting the inherent bias the estimator with an additive term (Qin and Priebe, 2017). However, the estimation of the location parameter of a symmetric distribution, such as normal distribution, is one of the few special cases for which the MLqE is consistent by itself. We proceed with using not-bias corrected MLqE, since there is no bias to correct for our parameter of interest, but we draw the readers' attention to the fact that an asymptotically unbiased estimator for σ^2 can be obtained by modifying the MLqE by a factor of q, $\hat{\sigma}^2_{corrected} = q\hat{\sigma}^2$

In our work we are mostly concerned with the normal distribution, most frequently used case in practice. Below we discuss in more details the three cases of LqRT with the normal distribution assumption.

2.3 One-sample test

Suppose we are interested in testing for the mean of a normal distribution, i.e., $H_0 : \mu = \mu_0$ versus $H_1 : \mu \neq \mu_0$. Let $f(\cdot|\mu, \sigma^2)$ represent the normal density. The test statistic of LqRT is

$$D_q(\boldsymbol{x}) = 2 \sup_{\substack{\mu \in \mathbb{R} \\ \sigma^2 \in \mathbb{R}^+}} \left\{ \sum_{i=1}^{n} L_q \left(f(x_i|\mu, \sigma^2) \right) \right\} - 2 \sup_{\sigma^2 \in \mathbb{R}^+} \left\{ \sum_{i=1}^{n} L_q \left(f(x_i|\mu_0, \sigma^2) \right) \right\}. \tag{2.1}$$

Note that LqRT becomes one-sample t-test when $q = 1$, so LqRT can be considered as a robustified version of t-test. To obtain the MLqE of the mean of a normal distribution and compute D_q, we use a special case of iterative reweighting algorithm outlined in Algorithm 2 to optimize the parameters for the first term and a known-mean version (Algorithm 5) for the second term. More specifically, Algorithm 2 determines an MLqE for both the mean and the variance of a sample that comes from a normal distribution. Such a algorithm is only applicable if there are no restrictions on the parameters.

The clipping of the variance in step 11 of Algorithm 2 is required in the implementation to avoid the division by 0 in the cases when the variance shrinks down unceasingly. The ϵ is usually chosen to be some very small number, for

Algorithm 2 Iterative re-weighting algorithm for an MLqE of a one sample from a one-dimensional normal with no additional constraints

function MLQE-NORMAL($\boldsymbol{x}$, q)
    $n \leftarrow$ LENGTH($\boldsymbol{x}$)

    # Initialize Parameters with MLE
    $\hat{\mu}^{(0)} \leftarrow n^{-1} \sum_{i=1}^{n} x_i$
    $\hat{\sigma}^{2^{(0)}} \leftarrow n^{-1} \sum_{i=1}^{n} (x_i - \hat{\mu}^{(0)})^2$

    # Iterative Re-weighting
    for $k = 1, 2, \ldots$ *until convergence* **do**

        # Update Weights
        for $i \leftarrow 1, 2, \ldots, n$ **do**
            $w_i^{(s)} \leftarrow \left(f \left(x_i | \hat{\mu}^{(s-1)}, \hat{\sigma}^{2^{(s-1)}} \right) \right)^{1-q}$
        end for

        # Update Parameters
        $\hat{\mu}^{(s)} \leftarrow \frac{\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} w_i}$
        $\hat{\sigma}^{2^{(s)}} \leftarrow \frac{\sum_{i=1}^{n} w_i (x_i - \hat{\mu}^{(s)})^2}{\sum_{i=1}^{n} w_i}$

        # Clip Variances
        if $\hat{\sigma}^{2^{(i)}} < \epsilon$ **then**
            $\hat{\sigma}^{2^{(i)}} \leftarrow \epsilon$
        end if
    end for
    return $\hat{\mu}^{(s)}, \hat{\sigma}^{2^{(s)}}$
end function

example, a numerical precision of the 64-bit floating point numbers in Python.

The shrinkage of the variance to 0 in the MLqE is a similar issue to the non-existence of the global MLE in the mixtures of Gaussians. In both of these situations, the estimating distribution becomes centered exactly at one of the data points, and the variance is allowed to decrease, which explodes the objective function, the likelihood in the mixtures of guassians or the Lq-likelihood in the Gaussian MLqE. The difference comes from the fact that in the mixtures case, this effect does not simultaneously affect the likelihood of all other points because the other components become responsible for them, whereas in MLqE, the weights of all points that are not centered exactly at the mean shrink as variance shrinks.

When performing the numerical experiments, we have observed this phenomenon predominantly in the bootstrapped samples used to estimated the p-value. Specifically, this effect tends to occur in samples where one observation is repeated a significant number of times. To give a sense of a scale, only about 0.2% of the bootstrapped samples converged to a degenerate distribution when the dataset size was 100. We have observed the shrinkage of the variance to 0 on the actual samples, as opposed to the ones bootstrapped by the testing procedure, but the occurrences of this were extremely sparse.

2.4 Two-sample paired test

Suppose that we have two samples $\boldsymbol{x} = [x_1, \ldots, x_n]^T$ and $\boldsymbol{y} = [y_1, \ldots, y_n]^T$, and the samples (x_i, y_i) are paired. For example, they correspond to the observations of the same patient in the beginning and in the end of some longitudinal study. The hypothesis to be tested is $H_0 : \mu_x = \mu_y$ against $H_1 : \mu_x \neq \mu_y$.

Similarly to the one sample test above, we can define a set of new variables $\boldsymbol{z} = [z_1, \ldots, z_n]$, such that $z_i = y_i - x_i$, and then perform a one-sampled test for $H_0 : \mu_z = 0$ against $H_1 : \mu_z \neq 0$, using the single sample procedure described above.

2.5 Two-sample unpaired test

Lastly, suppose we have two samples $\boldsymbol{x} = [x_1, \ldots, x_n]^T$ and $\boldsymbol{y} = [y_1, \ldots, y_m]^T$, not necessarily with the same size. The first sample comes from a normal distribution with density $f(x_i|\mu_x, \sigma_x^2)$ and the second from a normal distribution with a density $f(y_i|\mu_y, \sigma_y^2)$. We, once again, want to test for $H_0 : \mu_x = \mu_y$ against $H_1 : \mu_x \neq \mu_y$.

It is possible to do so with and without the shared variance assumption. In the former case we assume that $\sigma_x^2 = \sigma_y^2 = \sigma^2$, and can use the test statistic given by

$$\begin{aligned} D_q(\boldsymbol{x}, \boldsymbol{y}) =& 2 \sup_{\substack{\mu_x, \mu_y \in \mathbb{R} \\ \sigma^2 \in \mathbb{R}^+}} \left\{ \sum_{i=1}^{n} L_q \left(f(x_i|\mu_x, \sigma^2) \right) + \sum_{j=1}^{m} L_q \left(f(y_i|\mu_y, \sigma^2) \right) \right\} \\ &- 2 \sup_{\substack{\mu \in \mathbb{R} \\ \sigma^2 \in \mathbb{R}^+}} \left\{ \sum_{i=1}^{n} L_q \left(f(x_i|\mu, \sigma^2) \right) + \sum_{i=1}^{m} L_q \left(f(y_j|\mu, \sigma^2) \right) \right\} \end{aligned} \tag{2.2}$$

For the first term, we can combine the two samples together and use the regular reweighting algorithm (Algorithm 2) to determine the optimal parameters. For the second term, we have to use the re-weighting algorithm that estimates one shared variance, but two different means for two samples (Algorithm 6). This test procedure corresponds to a more robust version of the classical Student's t-test.

We can also make no assumption of the shared variances and obtain the test that is instead more similar in spirit to the Welch's t-test, but is more robust. Specifically, we use the test statistic given by

$$
\begin{aligned}
D_q(\boldsymbol{x}, \boldsymbol{y}) =& 2 \sup_{\substack{\mu_x, \mu_y \in \mathbb{R} \\ \sigma_x^2, \sigma_y^2 \in \mathbb{R}^+}} \left\{ \sum_{i=1}^{n} L_q \left(f(x_i | \mu_x, \sigma^2) \right) + \sum_{j=1}^{m} L_q \left(f(y_i | \mu_y, \sigma^2) \right) \right\} \\
& - 2 \sup_{\substack{\mu \in \mathbb{R} \\ \sigma_x^2, \sigma_y^2 \in \mathbb{R}^+}} \left\{ \sum_{i=1}^{n} L_q \left(f(x_i | \mu, \sigma^2) \right) + \sum_{i=1}^{m} L_q \left(f(y_j | \mu, \sigma^2) \right) \right\} \qquad (2.3)
\end{aligned}
$$

In order to optimize the first term, we use a constrained version of the re-weighting algorithm that estimates one shared mean for two samples, but two differnt variances (Algorithm 7). For the second term, there are no shared parameters, so it is sufficient to simply use the regular one-sampled re-weighting algorithm (Algorithm 2) on each of the two samples individually.

Modified versions of this algorithm that handle the case of two samples with a shared mean or variance, or the case when the mean is known a priori, are presented in the Appendix.

2.6 Estimating p-values

We have discussed how to compute the test statistic D_q, but so far made no statement on how to determine whether it falls inside the rejection region. In fact, the test statistic follows a scaled chi-square distribution. In practice, instead of obtaining the null distribution, we estimate the p-values.

In our implementation of the one-sample test, we estimate the p-value using the bootstrap procedure for the location parameter test provided in Qin and Priebe, 2017. This procedure relies on transforming the original sample

Algorithm 3 Bootsrap procedure to estimate the p-value for the one-sample LqRT.

function P-VALUE-1SAMPLE($\boldsymbol{x}$, u, q, B)
  $\hat{u}, \hat{\sigma}^2 \leftarrow$MLQE-NORMAL($\boldsymbol{x}$, q)
  $\boldsymbol{x}' \leftarrow [x_1 - \hat{u} + u, \ldots, x_1 - \hat{u} + u]^T$
  for $b = 1, \ldots, B$ **do**
    $\boldsymbol{x}'_b \leftarrow$ RESAMPLE($\boldsymbol{x}'$)
    $D_q^b \leftarrow D_q(\boldsymbol{x}'_b)$
  end for
  $D_q \leftarrow D_q(\boldsymbol{x})$
  $\widehat{p} \leftarrow$ quantile of D_q^bs that are greater than D_q
  return $\widehat{p}$
end function

Algorithm 4 Bootsrap procedure to estimate the p-value for the two-sample unpaired LqRT.

function P-VALUE-2SAMPLE-UNPAIRED($\boldsymbol{x}$, $\boldsymbol{y}$, q, B)
  $\hat{u}_x, \hat{\sigma}_x^2 \leftarrow$ MLQE-NORMAL($\boldsymbol{x}$, q)
  $\hat{u}_y, \hat{\sigma}_y^2 \leftarrow$ MLQE-NORMAL($\boldsymbol{y}$, q)
  $\boldsymbol{x}' \leftarrow [x_1 - \hat{u}_x, \ldots, x_1 - \hat{u}_x]^T$
  $\boldsymbol{y}' \leftarrow [y_1 - \hat{u}_y, \ldots, x_1 - \hat{u}_y]^T$
  for $b = 1, \ldots, B$ **do**
    $\boldsymbol{x}'_b \leftarrow$ RESAMPLE($\boldsymbol{x}'$)
    $\boldsymbol{y}'_b \leftarrow$ RESAMPLE($\boldsymbol{y}'$)
    $D_q^b \leftarrow D_q(\boldsymbol{x}'_b, \boldsymbol{y}'_b)$
  end for
  $D_q \leftarrow D_q(\boldsymbol{x}, \boldsymbol{y})$
  $\widehat{p} \leftarrow$ quantile of D_q^bs that are greater than D_q
  return $\widehat{p}$
end function

to be centered around the null hypothesis mean and then bootstrapping the distribution of D_q under the null. The exact algorithm for doing so is outlined in Algorithm 3.

For the two-sample unpaired test we propose a procedure that is very similar to the Algorithm 3, except both samples are centered around 0. This is accomplished by subtracting a robustly estimated means from the respective samples. The procedure is outlined in the Algorithm 4. This allows to bootstrap data from the null hypothesis under which both samples have the same means.

2.7 Selecting q

The tuning parameter q is important in our testing procedure because it controls the trade of between power and robustness, and it is generally not known a priori. In our approach, we select q by minimizing the trace of the asymptotic covariance matrix of $\widehat{\boldsymbol{\theta}}_q$. In a one-dimensional Gaussian case for a single sample this implies selecting

$$\begin{aligned}\hat{q} &= \arg\min_q \left\{ \hat{V}_q(\hat{\mu}_q) \right\} \\ &= \arg\min_q \left\{ a_q b_q a_q \right\}\end{aligned}$$

where

$$a_q = \left(\frac{1}{n} \sum_{i=1}^{n} \frac{\partial^2}{\partial \mu^2} L_q \left(f(x_i|\mu, \sigma^2) \right) \Big|_{\mu=\hat{\mu}_q, \sigma^2=\hat{\sigma}_q^2} \right)^{-1}$$

and

$$b_q = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{\partial}{\partial \mu} L_q \left(f(x_i|\mu, \sigma^2) \right) \Big|_{\mu=\hat{\mu}_q, \sigma^2=\hat{\sigma}_q^2} \right)^2$$

The $\hat{\mu}_q$ and $\hat{\sigma}_q^2$ are MLqE estimates over the whole hypothesis space. For a two-sample unpaired case, we select q in a similar fashion

$$\hat{q} = \arg\min_q \left\{ \hat{V}_q(\hat{\mu}_{y,q}) + \hat{V}_q(\hat{\mu}_{x,q}) \right\}$$

and the respective empirical variances are computed in a similar way to the one-dimensional case.

In our implementation, we use a grid search over the values of $q \in [0.5, 1.0]$ with an interval of 0.01 in order to solve this minimization problem. The smallest q is limited to 0.5, which corresponds to the minimum Hellinger distance estimation (Beran, 1977). The case of $q < 0.5$ is not yet well-studied (Qin and Priebe, 2017).

Chapter 3

Software

LqRT, the package that implements all of the proposed tests above, is available at https://github.com/alyakin314/lqrt. In this chapter we present the three core functions of **LqRT**, which correspond to the three versions of LqRT for testing the means of the normal distribution: one-sample, two-sample paired and two-sample unpaired.

3.1 One-sample test implementation

The one-sample Lq-likelihood-ratio-type test has the following function signature in the **LqRT**

```
lqrt.lqrtest_1samp(x, u,
                   q = None, bootstrap=100,
                   random_state=None)
```

The first two positional arguments, `x` and `u`, represent the sample observation and the expected value in null hypothesis in that order, similarly to `scipy.stats.ttest_1samp()`. However, unlike `scipy.stats.ttest_1samp()`

and other t-tests of **SciPy**, `lqrt.lqrtest_1samp()` does not support testing multiple samples or multiple hypotheses at once. Thus, the sample `x` must be one-dimensional array-like, and `u` must be a float. The returned object is a named tuple of the test statistic D_q and the p-value, likewise for all of other testes in the **LqRT**. This follows the convention of all the significance tests of **SciPy**.

The usage of the function is demonstrated in the example below. In this example, we first test whether the mean of a sample from a normal distribution is equal to its true value and then to a different value. We do not reject the null hypothesis in the first case and do reject in the second case.

```
>>> import lqrt
>>> import numpy as np
>>> from scipy import stats
>>>
>>> np.random.seed(314)
>>> rvs1 = stats.multivariate_normal.rvs(0, 1, 50)
>>>
>>> lqrt.lqrtest_1samp(rvs1, 0)
Lqrtest_1sampResult(statistic=0.02388120731922072, pvalue=0.85)
>>> lqrt.lqrtest_1samp(rvs1, 1)
Lqrtest_1sampResult(statistic=35.13171144154751, pvalue=0.0)
```

The optional argument `q` specifies the parameter q of the Lq-likelihood. The `q` typically should be within the interval [0.5, 1.0] and a lower value is associated with a more robust test. It can be specified manually or adaptively selected. The latter happens if it is set to `None` or is left unspecified. The adaptive selection procedures uses the trace of the empirical covariance procedure, outlined in Section 2.7. An example below demonstrates the usage of two different manually provided values for `q`, 0.9 and 0.6, as well as an adaptively selected one on the same data, generated from a gross-error model.

```
>>> rvs2 = np.concatenate(
...          [stats.multivariate_normal.rvs(0.32, 1, 45),
...           stats.multivariate_normal.rvs(0.32, 50, 5)])
>>>
>>> lqrt.lqrtest_1samp(rvs2, 0, q=0.9)

Lqrtest_1sampResult(statistic=2.239547159197258, pvalue=0.09)

>>> lqrt.lqrtest_1samp(rvs2, 0, q=0.6)

Lqrtest_1sampResult(statistic=3.4268748448623256, pvalue=0.02)

>>> lqrt.lqrtest_1samp(rvs2, 0)

Lqrtest_1sampResult(statistic=2.7337572196229587, pvalue=0.03)
```

The p-value is obtained via a bootstrap procedure, described in Section 2.6. The number of samples in a bootstrap can be varied using the `boostrap` argument to `lqrt.lqrtest_1samp()`. Increasing the number of samples increases the precision of the p-value, but adds on computational work. As a rough example, the three `lqrt.lqrtest_1samp()` calls below took 0.3s, 1.5s and 15s, respectively.

```
>>> lqrt.lqrtest_1samp(rvs1, 0, bootstrap=100)
Lqrtest_1sampResult(statistic=0.02388120731922072, pvalue=0.85)
>>> lqrt.lqrtest_1samp(rvs1, 0, bootstrap=1000)
Lqrtest_1sampResult(statistic=0.02388120731922072, pvalue=0.875)
>>> lqrt.lqrtest_1samp(rvs1, 0, bootstrap=10000)
Lqrtest_1sampResult(statistic=0.02388120731922072, pvalue=0.8743)
```

It should also be noted that the boostrapped resampling is random. The argument `random_state` allows to seed the random number generator, which allows reproducible results.

3.2 Two-sample paired test implementation

The two-sample LqRT Lq-likelihood-ratio-type test has the function signature

```
lqrt.lqrtest_rel(x_1, x_2,
                 q=None, bootstrap=100,
                 random_state=None)
```

where `x_1` and `x_2` are two samples, which must be array-like, one-dimensional and of equal size. This function is a wrapper for the `lqrt.lqrtest_1samp`. It extends the test from one-sample to a paired by subtracting one sample from the other within the pairs and setting the null hypothesis mean, `u`, to 0.

We provide an example of the `lqrt.lqrtest_rel()` usage. First, we use the test on two samples from a normal distribution which actually have identical population means:

```
>>> import lqrt
>>> from scipy import stats
>>> import numpy as np
>>> np.random.seed(314)
>>>
>>> rvs1 = stats.multivariate_normal.rvs(0, 1, 50)
>>> rvs2 = stats.multivariate_normal.rvs(0, 1, 50)
>>>
>>> lqrt.lqrtest_rel(rvs1, rvs2)

Lqrtest_relResult(statistic=0.22769245832813567, pvalue=0.66)
```

Now, we use the test on two samples drawn from the normal distributions with different means:

```
>>> rvs3 = stats.multivariate_normal.rvs(1, 1, 50)
>>> lqrt.lqrtest_rel(rvs1, rvs3)

Lqrtest_relResult(statistic=27.827284933987784, pvalue=0.0)
```

The parameters `q`, `bootstrap` and `random_state` work identically to the one-sample case, described in the previous section.

3.3 Two-sample unpaired test implementation

Lastly, we present a function that implements a two-sample unpaired LqRT

```
lqrtest_ind(x_1, x_2, equal_var=True,
            q=None, bootstrap=100,
            random_state=None)
```

The first two positional arguments, `x_1` and `x_2`, again, correspond to the two test samples. They must be array-like and one-dimensional, but need not be of the same size. The test can be run with or without the equal population variance assumption. This is done by varying the `equal_var` flag, similarly to the `scipy.stats.ttest_ind()`. When set to `True`, the test corresponds to a more robust version of the standard Student's t-test. When set to `False`, the test corresponds to a more robust version of the Welch's t-test. The default value of `equal_var` is `True`. We present the examples of using the unpaired test, both with and without the shared variance assumption.

```
>>> import lqrt
>>> from scipy import stats
>>> import numpy as np
>>> np.random.seed(314)
```

First, we generate two samples from a normal distribution that have different sizes but are both centered around 0. We test whether their means are the same with and without the equal variance assumption.

```
>>> rvs1 = stats.multivariate_normal.rvs(0, 1, 50)
```

```
>>> rvs2 = stats.multivariate_normal.rvs(0, 1, 70)
>>> lqrt.lqrtest_ind(rvs1, rvs2)

LqRtest_indResult(statistic=0.00046542438241203854, pvalue=0.99)

>>> lqrt.lqrtest_ind(rvs1, rvs2, equal_var=False)

LqRtest_indResult(statistic=0.00047040017227573117, pvalue=0.97)
```

Now, we generate a new sample from a normal distribution which has a different mean from the first two. We then use the LqRT to test for the equivalence of means of the first sample and the recently generated one. We do so both with and without the equal variance assumption.

```
>>> rvs3 = stats.multivariate_normal.rvs(1, 1, 70)
>>> lqrt.lqrtest_ind(rvs1, rvs3)

LqRtest_indResult(statistic=31.09168298440227, pvalue=0.0)

>>> lqrt.lqrtest_ind(rvs1, rvs3, equal_var=False)

LLqRtest_indResult(statistic=31.251454446588696, pvalue=0.0)
```

The parameters `q`, `bootstrap` and `random_state`, once again, work identically to the one-sample case.

3.4 Real data example

We also demonstrate a usage of the **LqRT** on the real data. We use Breast Cancer Wisconsin (Diagnostic) Data Set as an example. This data set can be easily import in Python using the **scikit-learn** package (Pedregosa et al., 2011). It is a copy of dataset located at the UC Irvine Machine Learning Repository ML (Dua and Graff, 2017).

The features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass, and summarized as the mean, standard error, and worst (largest) for each image, resulting in 30 total features per sample. We only use the features' means, which corresponds to the first ten dimensions. The dataset also includes binary labels corresponding to the tumor being malignant or benign. We present an example below in which we stratify the data on the true label and use a two-sample unpaired test on each of the ten features.

```
>>> import numpy as np
>>> from sklearn.datasets import load_breast_cancer
>>> np.random.seed(314)
>>>
>>> X, y = load_breast_cancer(return_X_y=True)
>>> X_negative = X[y==0]
>>> X_positive = X[y==1]
>>> features = 10
>>>
>>> for i in range(features):
...     print(lqrt.lqrtest_ind(X_positive[:, i],
...                            X_negative[:, i],
...                            equal_var=False,
```

```
...                                bootstrap=1000))

LqRtest_indResult(statistic=382.5469311314969, pvalue=0.0)
LqRtest_indResult(statistic=77.89690998094738, pvalue=0.0)
LqRtest_indResult(statistic=318.85934989217276, pvalue=0.0)
LqRtest_indResult(statistic=207.32949298709445, pvalue=0.0)
LqRtest_indResult(statistic=109.00139202641367, pvalue=0.0)
LqRtest_indResult(statistic=384.764767407446, pvalue=0.0)
LqRtest_indResult(statistic=758.937192422444, pvalue=0.0)
LqRtest_indResult(statistic=1313.77261746955, pvalue=0.0)
LqRtest_indResult(statistic=93.09648089232905, pvalue=0.0)
LqRtest_indResult(statistic=0.0749898025942457, pvalue=0.84)
```

Chapter 4

Experiments

We compare the performance of the **LqRT** with other popular tests implemented in Python, using the synthetically generated data.

In order to model the contamination in the data, we use a version of the gross error model (Huber, 1964) in which both the underlying true distribution and the anomalous values have a normal distribution. The two distributions are centered at the same location, but the one corresponding to the outliers is much wider. Take ϵ to represent the probability of observing a gross error, or an outlier. This corresponds to the density of the form

$$g(x_i|\mu, \sigma^2, \tau^2, \epsilon) = (1-\epsilon) f(x_i|\mu, \sigma^2) + \epsilon f(x_i|\mu, \tau^2). \tag{4.1}$$

where f is the density function of the normal distribution, $\epsilon < 0.5$ and $\sigma < \tau$. (Bickel and Doksum, 2006, page 357)

The ϵ is typically not known a priori. So, even though in all of the examples provided in this section the data is generated according to the density in Equation 4.1, the tests used assume a regular one-dimensional two-parameter normal distribution.

Test	Means (Size)		Means (Power)		Variances		
	μ_1	μ_2	μ_1	μ_2	σ_1^2	σ_2^2	τ
One-sample	0	-	0.34	-	1	-	50
Two-sample paired	0	0	0	0.50	1	1	50
Two-sample unpaired (equal variance)	0	0	0	0.50	1	1	50
Two-sampled unapired (no equal variance)	0	0	0	0.50	1	0.01	50

Table 4.1: Overview of the parameters used and the hypotheses tested in various set-ups for the gross error model simulation.

We generate the samples from this model. For all tests, the samples used had a size of 50. For the one-sample case, the hypotheses tested were $H_0 : \mu_1 = 0$ against $H_1 : \mu_1 \neq 0$. In all of the two-sample cases, the hypotheses were $H_0 : \mu_1 = \mu_2$ against $H_1 : \mu_1 \neq \mu_2$. Other parameters used in the simulations to estimate the power against an alternative are summarized in the Table 4.1. In the paired case we constrained the samples in the pair to either both have gross errors, or both not be such.

The performance of the **LqRT** was compared to that of other tests implemented in Python. The other tests used are listed in the Table 4.2. The results are presented in the Figure 4.1. Across all of the set-ups, the tests implemnted in the **LqRT** are valid, in the sense that the size is successfully controlled. Furthermore, in all of the four cases, the power of the Lq-likelihood-ratio-type tests dominates all other tests for large enough contamination.

Set-up	Tests used
One-sample	lqrt.lqrtest_1samp() scipy.stats.ttest_1samp() scipy.stats.wilcoxon() statsmodels.descriptivestats.sign_test()
Two-sample paired	lqrt.lqrtest_rel() scipy.stats.ttest_rel() scipy.stats.wilcoxon() statsmodels.descriptivestats.sign_test()
Two-sample unpaired (equal variance)	lqrt.lqrtest_ind(equal_var=True) scipy.stats.ttest_ind(equal_var=True) scipy.stats.ranksums()
Two-sampled unapired (no equal variance)	lqrt.lqrtest_ind(equal_var=False) scipy.stats.ttest_ind(equal_var=False) scipy.stats.ranksums()

Table 4.2: Overview of the tests used in different set-ups in the gross error model simulation.

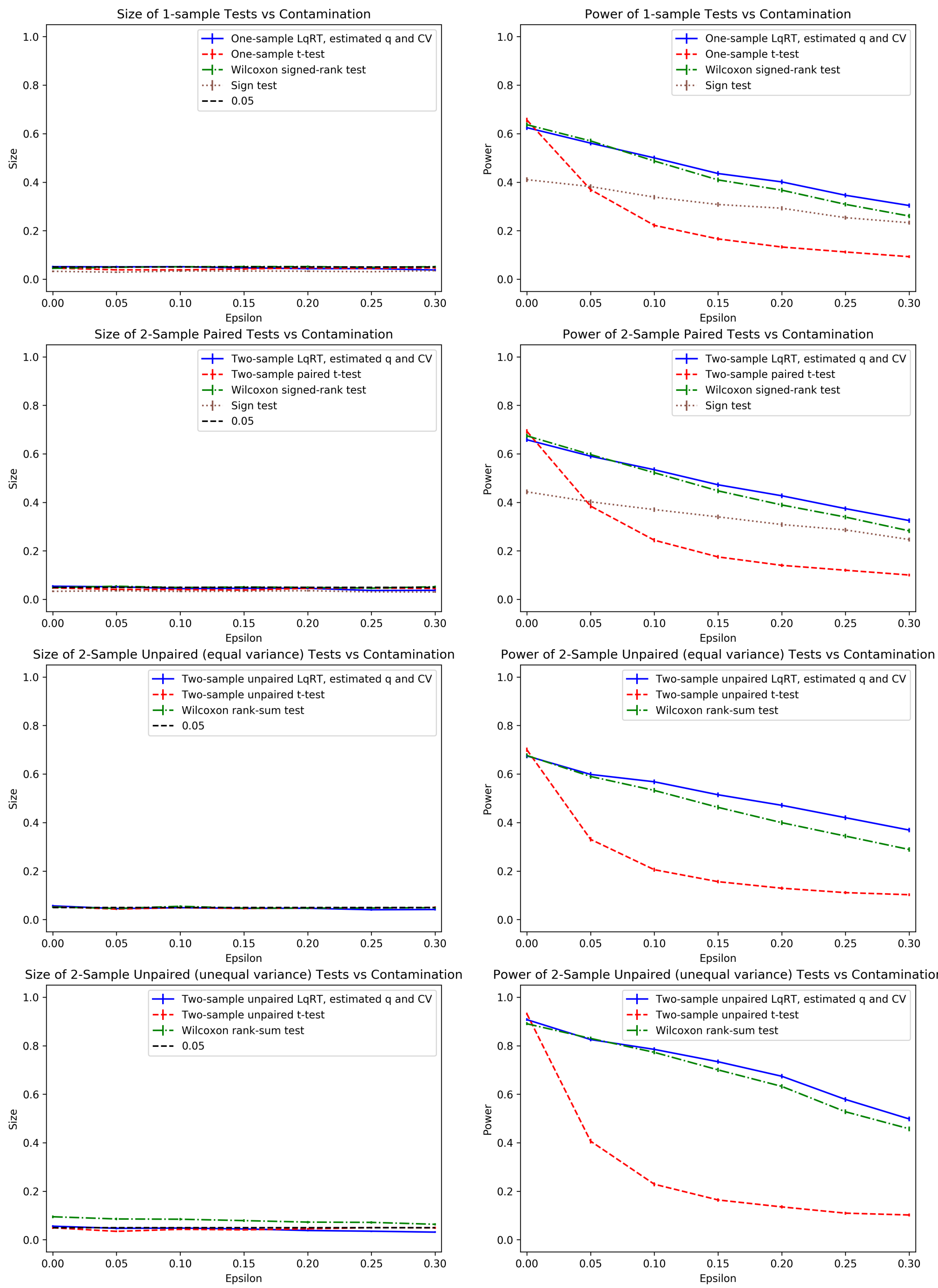

Figure 4.1: Gross error model simulation results. Each datapoint is produced using 10000 repetitions. The error bars represent 95% confidence interval.

Chapter 5

Conclusion and Future Work

In this work we have summarized the existing methodology of the Lq-likelihood-ratio-type test and introduced a way to generalize it to the two-sample unpaired Gaussian case.

We have also presented Python package, **LqRT**, offers an interface to use all of the cases one-dimensional Lq-likelihood-ratio-type tests. The package uses a syntax that closely resembles that of the statistical tests in the commonly used Python packages, such as **SciPy**. It has also been demonstrated on through simple examples that the tests implemented in the **LqRT**, including the two-sample unpaired case, are valid and perform as good or better than the competitors in terms of power when the data is contaminated.

The possible directions for the future work, both on the theoretical results of the Lq-likelihood-ratio-type tests and the implementations, include extending the Gaussian case to multiple dimensions, as well as implementing the tests for the distribution other than Gaussian. It is also possible to extend the package to the applications that commonly include the assumption of normality, such as significance test of the coefficients of the linear regression.

References

Grubbs, Frank E. (1969). "Procedures for Detecting Outlying Observations in Samples". In: *Technometrics* 11.1, pp. 1–21. DOI: 10.1080/00401706.1969.10490657. eprint: https://www.tandfonline.com/doi/pdf/10.1080/00401706.1969.10490657. URL: https://www.tandfonline.com/doi/abs/10.1080/00401706.1969.10490657.

Qin, Yichen and Carey E. Priebe (2017). "Robust Hypothesis Testing via Lq-Likelihood". In: *Statistica Sinica* 27, pp. 1793–1813.

Hampel, F.R., E.M. Ronchetti, P.J. Rousseeuw, and W.A. Stahel (2011). *Robust Statistics: The Approach Based on Influence Functions*. Wiley Series in Probability and Statistics. Wiley. ISBN: 9781118150689. URL: https://books.google.com/books?id=XK3uhrVefXQC.

Conover, W.J. (1999). *Practical nonparametric statistics*. Wiley series in probability and statistics: Applied probability and statistics. Wiley. ISBN: 9780471160687. URL: https://books.google.com/books?id=dYEpAQAAMAAJ.

Wilcoxon, Frank (1945). "Individual Comparisons by Ranking Methods". In: *Biometrics Bulletin* 1.6, pp. 80–83. ISSN: 00994987. URL: http://www.jstor.org/stable/3001968.

Mann, H. B. and D. R. Whitney (1947). "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other". In: *Ann. Math. Statist.* 18.1, pp. 50–60. DOI: 10.1214/aoms/1177730491. URL: https://doi.org/10.1214/aoms/1177730491.

Huber, Peter J. (1965). "A Robust Version of the Probability Ratio Test". In: *Ann. Math. Statist.* 36.6, pp. 1753–1758. DOI: 10.1214/aoms/1177699803. URL: https://doi.org/10.1214/aoms/1177699803.

Ferrari, Davide and Yuhong Yang (2010). "Maximum Lq-likelihood Estimation". In: *The Annals of Statistics* 38.2, pp. 753–783. DOI: 10.1214/09-AOS687. URL: https://doi.org/10.1214/09-AOS687.

Hodges, J. L. and E. L. Lehmann (1956). "The Efficiency of Some Nonparametric Competitors of the *t*-Test". In: *Ann. Math. Statist.* 27.2, pp. 324–335.

DOI: 10.1214/aoms/1177728261. URL: https://doi.org/10.1214/aoms/1177728261.

Python Software Foundation (2001–). *The Python Language Reference*. URL: https://docs.python.org/3/reference/.

Jones, Eric, Travis Oliphant, Pearu Peterson, et al. (2001–). **SciPy**: *Open source scientific tools for Python*. URL: http://www.scipy.org/.

Seabold, Skipper and Josef Perktold (2010). “**Statsmodels**: Econometric and statistical modeling with python”. In: *9th Python in Science Conference*.

Tsallis, Constantino (1994). “What are the numbers that experiments provide?” In: *Quimica Nova* 17, pp. 468–471. URL: http://submission.quimicanova.sbq.org.br/qn/qnol/1994/vol17n6/v17_n6_\%20(3).pdf.

Box, G. E. P. and D. R. Cox (1964). “An Analysis of Transformations”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 26.2, pp. 211–252. ISSN: 00359246. URL: http://www.jstor.org/stable/2984418.

Heritier, Stephane and Elvezio Ronchetti (1994). “Robust Bounded-Influence Tests in General Parametric Models”. In: *Journal of the American Statistical Association* 89.427, pp. 897–904. ISSN: 01621459. URL: http://www.jstor.org/stable/2290914.

Ferrari, Davide (2008). *Maximum Lq-likelihood Estimation*. PhD thesis. URL: https://conservancy.umn.edu/bitstream/handle/11299/60295/1/Ferrari_Davide\%20May\%202008.pdf.

Murphy, Kevin P. (2013). *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.]: MIT Press. ISBN: 9780262018029 0262018020. URL: https://www.amazon.com/Machine-Learning-Probabilistic-Perspective-Computation/dp/0262018020/ref=sr_1_2?ie=UTF8\&qid=1336857747\&sr=8-2.

Beran, Rudolf (1977). “Minimum Hellinger Distance Estimates for Parametric Models”. In: *Ann. Statist.* 5.3, pp. 445–463. DOI: 10.1214/aos/1176343842. URL: https://doi.org/10.1214/aos/1176343842.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). “**Scikit-learn**: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Dua, Dheeru and Casey Graff (2017). *UCI Machine Learning Repository*. URL: http://archive.ics.uci.edu/ml.

Huber, Peter J. (1964). “Robust Estimation of a Location Parameter”. In: *Ann. Math. Statist.* 35.1, pp. 73–101. DOI: 10.1214/aoms/1177703732. URL: https://doi.org/10.1214/aoms/1177703732.

Bickel, P.J. and K.A. Doksum (2006). *Mathematical Statistics 2e*. Pearson Education, Limited. ISBN: 9780131455924. URL: https://books.google.com/books?id=U1CCkgEACAAJ.

Appendix: Other Algorithms Used

There are three other iterative re-weighting algorithms that are employed in the **LqRT** in addition to the Algorithm 2. The pseudocode for all of them is provided below.

The Algorithm 5 only optimizes the variance of a sample. It is used in the one-sample version of the test under the null hypothesis assumption. It is also implicitly used by the two-sample unpaired test, since it wraps around the one-sample. The Algorithms 6 and 7 are both used when some parameter is optimized jointly. The former estimates two different means but one shared variance and is used in the two-sample unpaired test with the shared variance assumption. The latter optimizes the mean jointly, but the variances separately and is used in the two-sample unpaired test without the shared variance assumption.

Algorithm 5 Iterative Re-weighting algorithm for an MLqE of a one sample from a one-dimensional normal with a known mean

function MLQE-NORMAL-KNOWN-MEAN($\boldsymbol{x}$, u, q)
    $n \leftarrow$ LENGTH($\boldsymbol{x}$)

    # Initialize Parameters with MLE
    $\hat{\sigma}^{2^{(0)}} \leftarrow n^{-1} \sum_{i=1}^{n} (x_i - \mu)^2$

    # Iterative Re-weighting
    for $k = 1, 2, \ldots$ *until convergence* **do**

        # Update Weights
        for $i \leftarrow 1, \ldots, n$ **do**
            $w_i^{(s)} \leftarrow \left(f \left(x_i | \hat{\mu}^{(s-1)}, \hat{\sigma}^{2^{(s-1)}} \right) \right)^{1-q}$
        end for

        # Update Variance
        $\hat{\sigma}^{2^{(s)}} \leftarrow \frac{\sum_{i=1}^{n} w_i (x_i - \mu)^2}{\sum_{i=1}^{n} w_i}$
    end for

    return $\hat{\sigma}^{2^{(i)}}$
end function

Algorithm 6 Iterative Re-weighting algorithm for an MLqE of two samples from a one-dimensional normal with a shared variance constraint

function MLQE-NORMAL-2SAMPLE-EQUAL-VARIANCE($\boldsymbol{x}, \boldsymbol{y}, q$)
    $n \leftarrow$ LENGTH($\boldsymbol{x}$)
    $m \leftarrow$ LENGTH($\boldsymbol{y}$)

    # Initialize Parameters with MLE
    $\hat{\mu}_x^{(0)} \leftarrow n^{-1} \sum_{i=1}^{n} x_i$
    $\hat{\mu}_y^{(0)} \leftarrow m^{-1} \sum_{i=1}^{n} y_i$
    $\hat{\sigma}^{2^{(0)}} \leftarrow (n+m)^{-1} \left(\sum_{i=1}^{n} (x_i - \hat{\mu}_x^{(0)})^2 + \sum_{h=1}^{m} (y_j - \hat{\mu}_y^{(0)})^2 \right)$

    # Iterative Re-weighting
    for $k = 1, 2, \ldots$ *until convergence* **do**

        # Update Weights
        for $i \leftarrow 1, \ldots, n$ **do**
            $w_{x,i}^{(s)} \leftarrow \left(f\left(x_i | \hat{\mu}_x^{(s-1)}, \hat{\sigma}^{2^{(s-1)}} \right) \right)^{1-q}$
        end for
        for $j \leftarrow 1, \ldots, n$ **do**
            $w_{y,j}^{(s)} \leftarrow \left(f\left(y_j | \hat{\mu}_y^{(s-1)}, \hat{\sigma}^{2^{(s-1)}} \right) \right)^{1-q}$
        end for

        # Update Parameters
        $\hat{\mu}_x^{(s)} \leftarrow \frac{\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} w_{x,i}}$
        $\hat{\mu}_y^{(s)} \leftarrow \frac{\sum_{j=1}^{m} y_j}{\sum_{j=1}^{n} w_{y,j}}$
        $\hat{\sigma}^{2^{(s)}} \leftarrow \frac{\sum_{i=1}^{n} w_{x,i}^{(s)} (x_i - \hat{\mu}_x^{(s)})^2 + \sum_{j=1}^{n} w_{x,i}^{(s)} (y_j - \hat{\mu}_y^{(s)})^2}{\sum_{i=1}^{n} w_{x,i}^{(s)} + \sum_{j=1}^{n} w_{y,j}^{(s)}}$

        # Clip Variance
        if $\hat{\sigma}^{2^{(s)}} < \epsilon$ **then**
            $\hat{\sigma}^{2^{(s)}} \leftarrow \epsilon$
        end if
    end for

    return $\hat{\mu}_x^{(s)}, \hat{\mu}_y^{(s)}, \hat{\sigma}^{2^{(i)}}$
end function

Algorithm 7 Iterative Re-weighting algorithm for an MLqE of two samples from a one-dimensional normal with a shared mean constraint.

function MLQE-NORMAL-2SAMPLE-EQUAL-MEAN($\boldsymbol{x}, \boldsymbol{y}, q$)
    $n \leftarrow$ LENGTH($\boldsymbol{x}$)
    $m \leftarrow$ LENGTH($\boldsymbol{y}$)

    # Initialize Parameters with MLE
    $\hat{\mu}^{(0)} \leftarrow (n+m)^{-1} \left(\sum_{i=1}^{n} x_i + \sum_{i=1}^{m} y_i\right)$
    $\hat{\sigma}_x^{2^{(0)}} \leftarrow n^{-1} \sum_{i=1}^{n} (x_i - \hat{\mu}^{(0)})^2$
    $\hat{\sigma}_y^{2^{(0)}} \leftarrow m^{-1} \sum_{j=1}^{m} (y_j - \hat{\mu}^{(0)})^2$

    # Iterative Re-weighting
    for $k = 1, 2, \ldots$ *until convergence* **do**

        # Update Weights
        for $i \leftarrow 1, \ldots, n$ **do**
            $w_{x,i}^{(s)} \leftarrow \left(f\left(x_i | \hat{\mu}^{(s-1)}, \hat{\sigma}_x^{2^{(s-1)}}\right)\right)^{1-q}$
        end for
        for $j \leftarrow 1, \ldots, n$ **do**
            $w_{y,j}^{(s)} \leftarrow \left(f\left(y_j | \hat{\mu}^{(s-1)}, \hat{\sigma}_y^{2^{(s-1)}}\right)\right)^{1-q}$
        end for

        # Update Parameters
        $\hat{\mu}^{(s)} \leftarrow \frac{\sum_{i=1}^{n} w_{x,i}^{(s)} x_i + \sum_{j=1}^{m} w_{y,j}^{(s)} y_i}{\sum_{i=1}^{n} w_{x,i}^{(s)} + \sum_{j=1}^{m} w_{y,j}^{(s)}}$
        $\hat{\sigma}_x^{2^{(s)}} \leftarrow \frac{\sum_{i=1}^{n} w_{x,j} (x_i - \hat{\mu}^{(s)})^2}{\sum_{i=1}^{n} w_{x,i}}$
        $\hat{\sigma}_y^{2^{(s)}} \leftarrow \frac{\sum_{j=1}^{m} w_{y,j} (y_j - \hat{\mu}^{(s)})^2}{\sum_{j=1}^{m} w_{y,j}}$

        # Clip Variances
        if $\hat{\sigma}_x^{2^{(s)}} < \epsilon$ **then**
            $\hat{\sigma}_x^{2^{(s)}} \leftarrow \epsilon$
        end if
        if $\hat{\sigma}_y^{2^{(s)}} < \epsilon$ **then**
            $\hat{\sigma}_y^{2^{(s)}} \leftarrow \epsilon$
        end if

        return $\hat{\mu}^{(s)}, \hat{\sigma}_x^{2^{(s)}}, \hat{\sigma}_y^{2^{(s)}}$
    end for
end function

Curriculum Vitae

Anton Alyakin was born in Geneva, Switzerland on March 26, 1997. He graduated from TASIS The American School in Switzerland in 2015 and subsequently enrolled at Johns Hopkins University. While at Hopkins he studied Computer Science and Applied Mathematics and Statistics for his Bachelor of Science degree. He enrolled in the combined Bachelor's and Master's program in Applied Mathematics and Statistics where he focused on Statistics and Statistical Learning. He earned his Bachelor's and Master's degrees in May and December 2019, respectively. He currently researches statistical pattern recognition and machine learning techniques.